HOGESCHOOL ROTTERDAM	Tentamen **Computer Architecture**			Modulecode **ELECTA01**	
	Instituut	Engineering and Applied Science (EAS)			
	Opleiding	Elektrotechniek	**Docent**	BroJZ	**Assessor** BaRoy
	Datum	10-04-2018	**Duur**	135 minuten	

Normering	
$Cijfer = \frac{Punten}{7{,}5}$	Maximum: 75 punten. Aantal punten per vraag is aangegeven onder het opgavenummer.
Uitsluitend Toegestane Hulpmiddelen	
Rekenmachines: Geen programmeerbare rekenmachines toegestaan.	Boeken/dictaten/aantekeningen/formuleblad: Alleen Patterson & Hennessy ARM Edition met correcties is toegestaan, geen slides, geen aantekeningen, geen ander materiaal.
Overige opmerkingen	
Geef **altijd** je beredenering bij je antwoord.	
Naast deze zaken gelden de Procedures met betrekking tot Tentamens en Examens uit de studiegids.	

1
(6+3 = 9)
LD1

Gegeven is een wafer met een straal (*r*) van 15 cm. Deze wafer bevat 400 dies en heeft 0,035 defects/cm^2.

a) Bereken de yield volgens de formules die gegeven zijn op pagina 28 van je boek. Voor mensen die niet goed hebben opgelet op de middelbare school: de oppervlakte van een cirkel (wafer) is $\pi \cdot r^2$.
Die area = Wafer area / Dies per wafer = $\pi \cdot 15^2$ / 400 = 1,767 cm^2.
Yield = 1 / (1 + (Defects per area · Die area / 2))^2 = 1 / (1 + (0,035 · 1,767 / 2))^2 = 0,941

b) Bereken de kosten per die als gegeven is dat de kosten per wafer € 1200 zijn.
Cost per die = Cost per wafer / (Dies per wafer · Yield) = 1200 / (400 · 0,941) = € 3,19

2
(4+2 = 6)
LD1

Gegeven is een programma met een executietijd (*t*) van 64 s op een systeem met één processor. Als we dit programma op *p* processoren uitvoeren is gegeven dat de executietijd gelijk is aan *t*/*p* s + 4 s aan overhead. Deze overhead is onafhankelijk van het aantal processoren.

a) Bereken de executietijd van dit programma bij het gebruik van 4, 16 en 64 processoren.
executietijd voor p == 4 = 64 / 4 + 4 = 20 s
executietijd voor p == 16 = 64 / 16 + 4 = 8 s
executietijd voor p == 64 = 64 / 64 + 4 = 5 s

b) Bereken de speedup die bereikt wordt door het gebruik van 4, 16 en 64 processoren t.o.v. het gebruik van één processor?
speedup voor p==4 = 64 / 20 = 3,2
speedup voor p==16 = 64 / 8 = 8
speedup voor p==64 = 64 / 5 = 12,8

3
(10)
LD2 en LD3

Zet de onderstaande C code om in LEGv8 assemblercode. Ga ervan uit dat de variabelen `i` en `a` zich bevinden in respectievelijk X0 en X1 en dat het basisadres van de array `b` zich bevindt in register X2. Alle variabelen en adressen zijn 64-bit (`long long` in C).

```
    for (i = 1; i < a; i++)
        b[i-1] = b[i-1] + b[i];
```

```
    ADDI    X0, XZR, #1
LOOPI:
    SUBS    XZR, X0, X1 of CMP X0, X1
    B.GE    ENDI
    LSL     X9, X0, #3
    ADD     X10, X2, X9
```

```
    LDUR    X11, [X10, #-8]
    LDUR    X12, [X10, #0]
    ADD     X13, X11, X12
    STUR    X13, [x10, #-8]
    ADDI    X0, X0, #1
    B       LOOPI
ENDI:
```

4
(10)
LD4

Zet het decimale getal 51,390625 om naar IEEE 754 single percision format en IEEE 754 double percision. Geef de antwoorden in hexadecimale notatie.
51,390625 = 110011,011001 = 1,10011011001 * 2^5
Single percision: s = 0, f = 10011011001 exp+bias = 5 + 127 = 132 = 1000100
0100.0010.0100.1101.1001.0000.0000.0000 = 0x424D.9000
Double precision: s = 0, f = 10011011001 exp+bias = 5 + 1023 = 1028 = 10000000100
0100.0000.0100.1001.1011.0010.0000.0000.0000.0000.0000.0000.0000.0000.0000.0000 = 0x4049.B200.0000.0000

5
(3+3 = 6)
LD5

Bij de productie van de processor gegeven in figuur 4.17 op pagina 277 van je boek kan een fout optreden waardoor een signaal altijd hoog is. Dit wordt een “stuck-at-1 fault” genoemd.

a) Welke typen instructies (R, I, D en/of B) werken niet meer correct als het MemToReg signaal altijd hoog is?
R en I werken niet meer correct omdat niet de output van de ALU maar een willekeurige waarde in het destination register wordt geschreven.
b) Welke typen instructies (R, I, D en/of B) werken niet meer correct als het ALUSrc signaal altijd hoog is?
R werken niet meer omdat niet Rm maar een deel van de instructie als tweede argument gebruikt wordt.

6
(3+3+3 = 9)
LD5

De instructie 0x8A160360 wordt uitgevoerd door de eenvoudige implementatie van de LEGv8 processor gegeven in figuur 4.17 op pagina 277 van je boek. Elk X*n* register bevat de waarde *n*. Dus register X0 bevat de waarde 0, X1 de waarde 1 enz.
0x8A160360 = 10001010000.10110.000000.11011.00000 = AND X0,X27,X22

a) Wat is de waarde van het Reg2Loc control signal en wat is de waarde aan de output van de multiplexer die door dit signaal wordt aangestuurd?
0 uitgang mux = 10110
b) Wat is de waarde van het ALUSrc control signal en wat is de waarde aan de output van de multiplexer die door dit signaal wordt aangestuurd?
0 uitgang mux = inhoud van X22 = 22 = 0x0000.0000.0000.0000.0000.0000.0001.0110
c) Wat is de waarde van het MemtoReg control signal en wat is de waarde aan de output van de multiplexer die door dit signaal wordt aangestuurd?
0 uitgang mux = inhoud X22 AND inhoud X27 = 11011 AND 10110 = 10010 = 18 = 0x0000.0000.0000.0000.0000.0000.0001.0010

7
(6)
LD6

Gegeven is de pipelined implementatie van de LEGv8 processor gegeven in figuur 4.50 op pagina 315 van je boek. Deze processor heeft geen hardware om data hazards correct af te handelen. De compiler moet indien nodig NOP instructies toevoegen in de code. Waar **moet** de compiler NOP instructies toevoegen in de onderstaande code.

```
    ADDI X1, X2, #5
```

```
NOP
NOP
ADD  X3, X1, X1
ADDI X4, X1, #15
NOP
ADD  X5, X3, X1
```

8 (2+7 = 9) LD6

Een branch instructie die zich in een oneindige loop bevindt heeft het volgende gedrag: T, NT, T, T, T, T, NT, NT, NT, NT. Daarna herhaald dit patroon zich weer. T staat voor Taken en NT voor Not Taken.

a) Welk percentage van de branch instructies wordt correct voorspeld door een always-taken predictor.
5 / 10 = 50%

b) Welk percentage van de branch instructies wordt correct voorspeld door een 2-bit branch predictor? Zie figuur 4.62 op pagina 334 van je boek. Ga ervan uit dat de predictor zich in de toestand links onder bevindt als de loop start.
Toestanden nummer van linksboven naar linksonder. Predictor start in toestand 4.
Dus state:prediction:branch: 4:NT:T, 3:NT:NT, 4:NT:T, 3:NT:T, 2:T:T, 1:T:T, 1:T:NT, 2:T:NT, 3:NT:NT, 4:NT:NT dus 5/10 = 50%

9 (3+3 = 6) LD7

We beschouwen een processor met de volgende eigenschappen.

- Base CPI zonder memory stalls = 1,5
- Processor clock frequency = 2 GHz
- Main memory access time = 100 ns
- First-level cache miss rate per instruction (code and data misses combined) = 7%
- Second-level cache access time = 12 clock cycles
- Second-level miss rate per access (code and data misses combined) = 3,5%

a) Bereken de CPI voor de bovenstaande processor met **alleen** een first-level cache (dus **zonder** second-level cache).
Clock period = 1 / 2 GHz = 0,5 ns dus memory access = 100 / 0,5 = 200 clock cycles.
CPI = 1,5 + 0,07 * 200 = 15,5

b) Bereken de CPI voor de bovenstaande processor met **zowel** een first-level cache als een second-level cache.
CPI = 1,5 + 0,07 * (1 - 0,035) * 12 + 0,07 * 0,035 * 200 = 2,8006

10 (2+2 = 4) LD7

Een **volledig associatief** cache geheugen bevat 2 KiB aan data. De cache bestaat uit blokken die elk 256 bytes bevatten. De adressen zijn 64 bits breed.

a) Hoeveel comperatoren (om tags te kunnen vergelijken) bevat dit cache geheugen?
Aantal cache blocks = 2048 / 256 = 8. Volledig associatief dus 8 comperatoren.

b) Hoeveel bits breed zijn deze comperatoren?
Tag size = 64 – $^{2}\log 256$ = 64 – 8 = 56 bits dus comperatoren zijn 56 bits breed.